An Artificial Neural Network for Wavelet Steganalysis*

Clifford Bergman and Jennifer Davidson
Iowa State University

October 1, 2004–September 31, 2006

Abstract

Hiding messages in image data, called *steganography,* is used by criminals and noncriminals alike to send information over the internet. The detection of hidden messages in image data stored on websites and computers, called *steganalysis,* is of prime importance to cyber forensics. Automated detection of hidden messages is a requirement, since the shear amount of image data available online makes it impossible for a person to investigate each image separately. The purpose of this project is to develop a prototype software system that automatically classifies an image as having hidden information or not, using a powerful classifier called an artificial neural network (ANN). The novelty of this ANN is its ability to detect messages hidden with wavelet embedding algorithms, in addition to other transforms.

1 Project Description

Steganalysis is of increasing importance to cyber security. While the number of freeware packages available for steganography is increasing each year, the detection of most of these methods is neither satisfactory nor fully automated. While it is possible to hide messages within a variety of data file types, image data is likely to be the medium of choice for cyber criminals for several reasons. First, because of the high level of redundancy in image data, it is possible to embed a great deal of hidden information. Second, innocuous-looking images are commonplace on every computer and arouse little suspicion. Virtually all computer-users keep digital photos of friends and family, vacations, special events, etc. on their hard drives. Many web sites use images as a way to add interest and break up the monotony of

*Final Report to Midwest Forensics Resource Center

text. By contrast, audio or video files posted on web sites are prone to be examined for copyright infringement.

The shear volume of image data available online makes it difficult to identify suspicious content. Thus, automated stego detection systems that can accurately detect hidden data can bring great benefits to the cyber forensic community in terms of quick and accurate detection. After a suspicious image is detected, further processing can be performed to try to extract and decode the hidden message.

The primary result of this project is a prototype software package that takes an image as input and returns a classification for that image. Images will be classified as to whether or not they contain hidden information. Our project does not extract nor decode a hidden message, but simply determines if there is one embedded. The software package includes code for feature extraction from image data, code for an ANN classifier that uses the features, plus a Graphical User Interface (GUI) that enables users to easily use both those components. One novel feature of our system is the ability to detect wavelet-based steganography, which has not yet been done (according to an extensive literature review). Since jpeg2000 will be the new image compression standard in the near future, and jpeg2000 is based on the wavelet transform, our system puts forensic science techniques ahead of those of the criminals, instead of playing catch-up as is usually the case. The ANN is also designed to detect images embedded with other commonly-used steganography algorithms, in particular those that leave signatures in the transform domain, specifically the freeware applications `F5` and `jsteg`.

1.1 Steganography and Steganalysis

Steganography is a set of techniques for transmitting information from a sender (Alice) to a recipient (Bob) in such a way that an adversary (Warren) is unaware that any information is being sent. The original concept dates back to Simmons [18]. It is important to distinguish steganography from the related field of *cryptography.* Imagine for a moment that Alice is a spy operating in enemy territory, and Bob is her contact at home. Alice has uncovered the plans for a new weapon that Warren is building. Alice could encrypt those plans before sending them to Bob. Warren, who is monitoring the internet, would intercept the message, but would not be able to read it. However, Warren would be aware that Alice has sent something obviously sensitive, and that would compromise her role as a spy.

Instead, Alice would try to conceal the very existence of the message from Warren. Historically, tricks such as invisible ink, tiny puncture marks in a piece of paper or documents hidden inside a physical object have been used for this purpose. In the electronic age, Alice might resort to digital hiding techniques. She could take a digital photo of herself "on vacation," and make some small modifications to the

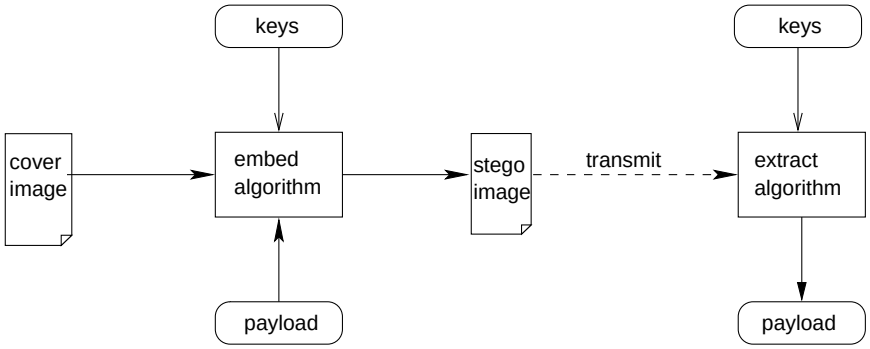

Figure 1: Communication model for steganography and steganalysis

resulting data file. The modifications would be minor enough to not impact the visual appearance of the photo, but would, nevertheless, contain the message Alice wishes to send. Bob, upon receipt of the photo, knows how to extract the message Alice wished to send. Warren, who has intercepted the transmission, would see nothing but an innocuous-looking photo of a lady on vacation. Figure 1 illustrates the communication process in which Alice and Bob engage.

Unfortunately, these same techniques can be used for ill. Criminals can hide incriminating data files (spreadsheets, documents, pornography) on their computer by concealing them within innocent images. There is some evidence that terrorists have used steganographic techniques to convey instructions to "sleeper cells" in the West [15]. Several pieces of data-hiding software are freely available for download, see for example [20]. A further introduction to this field can be found in [16].

For these reasons, there is (or should be!) interest from the counterterrorism and law-enforcement communities in measures that can be used to detect the existence of hidden data. This is steganalysis. There are two basic approaches to steganalysis. One method is to simply apply every known steganographic algorithm to the file under investigation and see whether the extracted message "makes sense". Unfortunately, any good steganography program will first encrypt the message before embedding it, and there is typically no way to distinguish an encrypted message from the random noise that results from applying the extraction algorithm to a truly innocent file.

Alternatively, one can use statistical methods to distinguish between innocent and steganographic files. This has the potential advantage of being more general: different steganographic algorithms may result in similar statistical "anomalies". For example, in a typical jpeg image, the AC coefficients are much more likely to be even numbers than to be odd. But after a naïve steganographic technique

called LSB encoding, these coefficients will be equally likely to be even or odd. The discrepancy between even and odd coefficients can be captured by a statistical quantity. These quantities are collectively called *features.*

A single feature may provide only scant indication of the presence of steganography, or, several features may on their face conflict in their diagnosis. What is needed is a method of combining multiple features into a single conclusion of "stego" or "innocent". For this we utilize a pattern recognition system called an *artificial neural network* (ANN).

Developing an ANN is a two-stage process. First the network is *trained* by feeding it the features from a large pool of images, some of which are known to contain stego, and some that are known to not contain stego. Based on the training, the neural net determines computational rules that can then be applied to the features of an image of unknown character.

The particular ANN software utilized in this project [2] is also able to select a subset of the original feature set consisting of those most useful for the characterization task at hand. This has the advantage of speeding up the final software package. In our case, the initial collection of 81 statistical features was culled to 22 that were judged to be most promising.

A necessary byproduct of our project is a large database of steganographic images. As described in Section 3, we built a set of images containing data hidden with several different steganographic algorithms at several different capacities. This database will undoubtedly prove useful for further research by our team as well as others.

1.2 Terminology

Let us define the terms used in Figure 1. The purpose of steganography is to hide sensitive information in an otherwise innocent computer file. This file is called the *cover.* In this paper all covers are assumed to be graphic files, so naturally, we shall refer to them as cover images. The information to be hidden is called the *payload.* The cover image and payload are combined by an *embedding algorithm,* producing a *stegoimage.* We prefer the term payload to "message" since the latter term seems a bit ambiguous. After all, one might construe "message" to refer to the entire stegoimage (which is what gets sent) rather than just the hidden data.

At the receiving end, the payload must be *extracted* from the stegoimage. As a rule, the original cover image has no value for the recipient, so it is discarded. The embedding algorithm may or may not require additional inputs, called *keys.* These keys might be used to encrypt the payload or might provide details as to how precisely the data is embedded. The recipient must know these keys in order to extract the payload.

The objective of steganalysis is to determine whether data-hiding is taking place. An image containing hidden information is called a stegoimage. An image with no hidden data will be called *innocent.*

2 Objectives

The primary objective of this project was to produce a prototype software system that could be used to detect the presence of hidden data in otherwise innocent-looking image data files. The software was to have an easy-to-use graphical user interface and be flexible enough to detect different steganographic techniques including those operating in both the DCT and wavelet domains.

We call the package ANNTS (artificial neural network technology for steganalysis). ANNTS is usable by law-enforcement personnel in its present form, but more important, it demonstrates the feasibility of using artificial neural nets to detect the presence of steganography in both conventional jpeg as well as jpeg2000-format images. It should be pointed out that the software only detects, and does not attempt to extract or decode the hidden data. Obviously extraction of the data is desirable, however successful extraction would involve the ability to break certain ciphers that are considered to be quite strong. We hope that the software will prove useful as it is, allowing law-enforcement to focus its efforts on those images (and users) that test positive.

3 Procedures

The project involved several distinct steps.

1. Build database of stego image files;
2. Choose features and implement code to extract features from images;
3. Design and train artificial neural nets;
4. Implement package and test.

3.1 The Steganographic Database

We began with a set of 1,300 "clean" images in color png (a bitmapped) format provided to us by Dr. Ed Delp of Purdue University [8]. For our purposes, each image was converted to two other forms: pgm (greyscale bitmap) and bmp (color bitmap). The reason for these multiple formats is that different steganographic algorithms operate on images in different formats. Each of three steganographic algorithms was applied to every one of the images with random data embedded at both 50% and 100% of capacity. In addition, the bitmapped images were converted

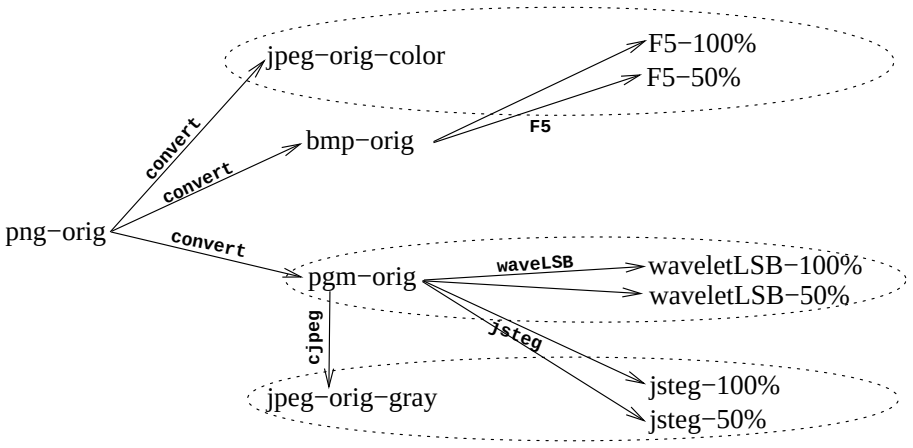

Figure 2: Conversion and processing of image data

to jpeg (with no data embedded) to act as the innocent set for training purposes. A flowchart illustrating the complete set of conversions is given in Figure 2.

We chose two freely available algorithms as the basis for our tests, `jsteg` [20] and `F5` [21]. `jsteg` takes as input an image in a bitmapped format and returns a stegogram in jpeg format. `F5` takes as input a color image in bmp format and returns a stegogram as a color jpeg.

We intended our third algorithm to be one that operated in a wavelet domain. Unfortunately, no such algorithms seem to be publicly available at this time. So we wrote our own. The algorithm works by inserting payload bits in the least significant bit position of each wavelet coefficient. Details are in Appendix B.

We should point out that our wavelet steganographic algorithm is not a good one. Upon extraction, a high percentage of the payload bits will be incorrect. This is due to the action of round-off and clipping. However, we believe the algorithm is adequate for our purposes: a procedure that makes changes in the wavelet domain that are visually imperceptible in the spatial domain.

3.2 Feature Selection and Extraction Code

After an extensive literature review, we settled on an initial set of 81 statistical features. These were coded and the statistics for all images in the database were computed. Based on the results of the ANNs, a final subset of 22 features were selected for use in the software. Feature extraction was implemented in Matlab 7.0.1. A discussion of the specific features follows. The list of 22 final features is given

in Appendix C.

Moments of the wavelet transform

The wavelet transform is a multi-scale transformation of image data that can be viewed as giving spatial location of frequency content. Many natural-scene image data display strong correlations between the scales in a multi-scale decomposition, see for example [11]. Farid in [9] exploited the nature of these statistical regularities when he designed (three different) Fisher linear discriminant analyzers to detect steganographic content in images using three different embedding algorithms. Our intent was to use a subset of Farid's 72 features in our single nonlinear ANN (along with features from other sources) to classify images possibly containing hidden data produced by several different embedding schemes.

The 72 features fall into two groups of 36 each, all of which are computed from the 4-scale Haar wavelet coefficients. A brief description of the transformation process is given in Appendix B. The first group of 36 features, consists of the mean, variance, skewness and kurtosis computed at each of the first 3 levels on each subband image. Note that at each level, there are 3 subband images, so we obtain $3 \cdot 3 \cdot 4 = 36$ feature values.

The remaining 36 feature values come from the log errors in the optimal linear predictor of coefficient magnitude. The linear coefficient predictor we used is based on "seven important neighbors," see [6] for example. For the vertical subband, the coefficient $V_i(x, y)$ can be predicted in a linear fashion by

$$\begin{aligned} V_i(x,\, y) =& w_1 V_i(x-1,\, y) + w_2 V_i(x+1,\, y) + \\ & w_3 V_i(x,\, y-1) + w_4 V_i(x,\, y+1) + w_5 V_{i+1}(x/2,\, y/2) + \\ & w_6 D_i(x,\, y) + w_7 D_{i+1}(x/2,\, y/2), \end{aligned} \tag{1}$$

where the w_i's are the unknown scalar weighting values, i denotes the scale or level value, and (x, y) denotes the pixel location in the image subband. Graphically, we can denote the nearest neighbors of the pixel $V_i(x, y)$ as in Figure 3. There is a similar relation for the diagonal and horizontal subbands:

$$\begin{aligned} H_i(x,\, y) =& w_1 H_i(x-1,\, y) + w_2 H_i(x+1,\, y) + \\ & w_3 H_i(x,\, y-1) + w_4 H_i(x,\, y+1) + w_5 H_{i+1}(x/2,\, y/2) + \\ & w_6 D_i(x,\, y) + w_7 D_{i+1}(x/2,\, y/2) \end{aligned}$$

$$\begin{aligned} D_i(x,\, y) =& w_1 D_i(x-1,\, y) + w_2 D_i(x+1,\, y) + \\ & w_3 D_i(x,\, y-1) + w_4 D_i(x,\, y+1) + w_5 D_{i+1}(x/2,\, y/2) + \\ & w_6 H_i(x,\, y) + w_7 V_i(x,\, y). \end{aligned}$$

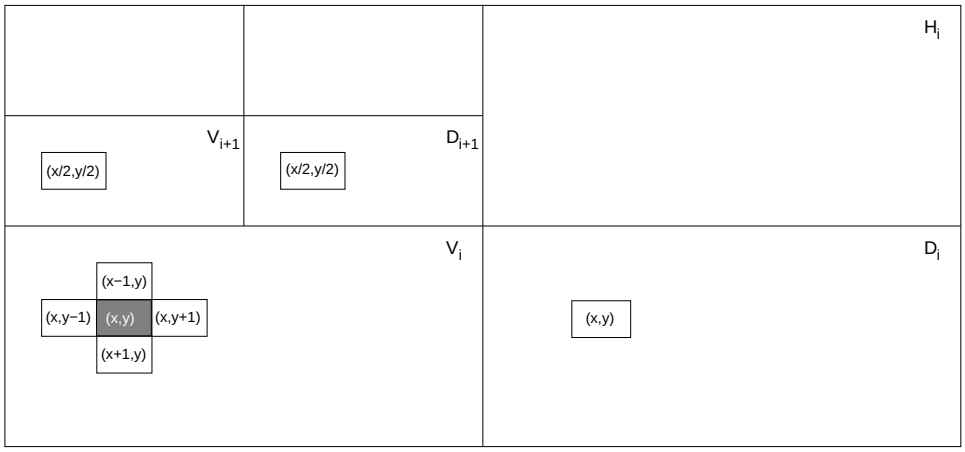

Figure 3: Illustration of nearest neighborhood pixels used in equation (1). The shaded box represents $V_i(x, y)$.

For each subband type (vertical, diagonal, and horizontal), the expression can be written in matrix form. For example, for the vertical subband, we write

$$\mathbf{V} = Q\mathbf{w}$$

and then the unknown variables $\mathbf{w}$ are solved for, using the least mean square solution

$$\mathbf{w} = (Q^T Q)^{-1} Q^T \mathbf{V}.$$

Finally, the log error in this predictor is given by

$$\mathbf{E}_V = \log_2 \mathbf{V} - \log_2 |Q\mathbf{w}|.$$

The log error is a vector, and from this vector are computed the same moments as before: mean, variance, skewness, kurtosis, yielding $3 \cdot 3 \cdot 4$ additional features, for a total of 72 features altogether.

Image quality metrics

Image quality assessment is an active research area whose goal is to create an objective measure that "correlates well" with a subjective human assessment of the image. Image quality metrics (IQMs), as they are sometimes called, have been used extensively, for example, in evaluating compression algorithms for image and

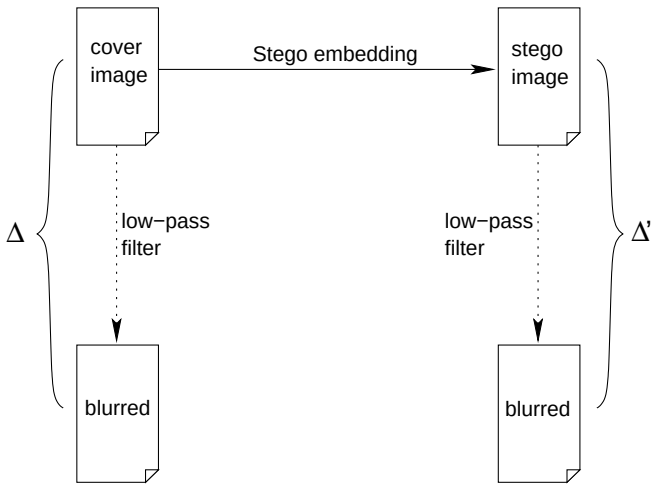

Figure 4: Strategy behind use of IQM

video data. They provide a reference by which different compression algorithms applied to many different images or videos can be compared side-by-side.

The IQMs used for steganalysis here were taken from [1]. The expectation is that the disruption to an image caused by low-pass filtering will be different for an innocent image as compared to a stegoimage. Figure 4 attempts to illustrate this idea. Δ represents the "difference" between an image and its blurred counterpart and can be used as a feature. The expectation is that if we begin with an innocent cover image and embed data in it, the vales of Δ and Δ' will be sufficiently far apart to be measurable.

To be more precise, a Gaussian filter, H (see equation (2)), is applied to a candidate image, I, producing a blurred version of the image, $H(I)$. An IQM, μ, is used to measure the difference between the image and its blurred counterpart, yielding a quantity $\Delta = \mu(I, H(I))$. By training the neural network on values of Δ obtained from both unmodified and stego images, we hope the neural net will later be able to distinguish between the two.

We selected three IQMs for this project. They are the median block spectral phase, the median weighted block spectral distortion and the normalized mean square HVS error. A description of these metrics is given in equations (3)–(5) below.

To define the filter and the image quality metrics we assume our image is a rectangular, single-band array $I_{i,j}$, $0 \leq i \leq m-1$, $0 \leq j \leq n-1$. The blurred

image $H(I)$ is obtained by convolving I with the 3×3 kernel ηG where

$$G_{uv} = \frac{2}{\pi} \exp\bigl(-2(u^2+v^2)\bigr), \qquad \text{for } -1 \le u, v \le 1 \text{ and}$$
$$\eta = \Bigl(\sum_{u,v=-1}^{1} |G_{uv}|^2 \Bigr)^{-1/2} \tag{2}$$

Let I and J be images of identical dimensions. The value of $\mu(I, J)$ where μ is one of the two block spectral metrics is computed as follows. I and J are each partitioned into blocks of size 32×32, enumerated as $I^{(k)}$, $J^{(k)}$, $k = 1, \ldots K$. For a block $I^{(k)}$, let $\hat{I}^{(k)}$ be its discrete Fourier transform. Then the *median block spectral phase difference* between I and J is defined as

$$\mu_1(I, J) = \operatorname*{median}_{k=1\ldots K} \sqrt{\sum_{i,j=0}^{31} \Bigl(\bigl|\arg(\hat{I}_{ij}^{(k)})\bigr| - \bigl|\arg(\hat{J}_{ij}^{(k)})\bigr| \Bigr)^2} \tag{3}$$

where $\arg(z)$ denotes the phase angle of the complex number z. Simiilarly, the *median weighted block spectral distortion* is given as

$$\lambda \cdot \mu_2(I, J) + (1 - \lambda) \cdot \mu_1(I, J). \tag{4}$$

Here $\mu_2(I, J)$ is defined in the same manner as $\mu_1(I, J)$, but with $\arg(z)$ replaced by $|z|$. The value of λ was chosen experimentally to be 0.000125.

Finally, the *normalized mean square Human Visual System metric* is defined as

$$\mu_3(I, J) = \frac{\|U(I) - U(J)\|_F^2}{\|U(I)\|_F^2} \tag{5}$$

where $\| \cdot \|_F$ denotes the Frobenius norm and $U(I) = D^{-1}\bigl(H \cdot D(I)\bigr)$. In this equation, the product is the Hadamard product, $D(I)$ denotes the discrete cosine transform of I, $\rho = \sqrt{u^2 + v^2}$, and $H_{u,v} = F(\rho)^2$ where

$$F(\rho) = \begin{cases} .05 \exp(\rho^{0.554}) & \text{for } \rho < 7 \\ \exp(-9(|\log \rho - \log 9|)^{2.3}) & \text{for } \rho \ge 7. \end{cases}$$

See [1] or [14] for more details.

Balanced parity

In [22] Westfeld and Pfitzmann observed that when encrypted data is embedded as the least significant bit in the bytes of a data file, those bytes will be equally

likely to be either even or odd. By contrast the bytes of a typical image file do not seem to have uniformly distributed parity. This seems to be true whether the bytes represent either pixel values or jpeg coefficients. Westfeld and Pfitzmann used this idea to design a statistical measure that could be used to distinguish stegoimages created with an LSB-based algorithm from innocent images. The technique was refined into a statistical feature in [13].

Let J be a portion of a fixed jpeg image, and let x_i denote the number of DCT coefficients in J equal to i for $i = -128, \ldots, 127$. Furthermore, define $x^*_{2i} = \frac{1}{2}(x_{2i} + x_{2i+1})$, for $i = -64, \ldots, 63$. We now take the statistic

$$\chi^2 = \sum_{\substack{i=-64 \\ i \neq 0}}^{63} \frac{(x_{2i} - x^*_{2i})^2}{x^*_{2i}} \tag{6}$$

which is a χ^2 statistic with 127 degrees of freedom. (If the values of x_{2i} or x^*_{2i} are less than 4 then equation (6) must be modified slightly. See [13] for details.)

The *balanced parity feature* for an image I is the maximum value of χ^2 as J ranges from 1%, 2%,...,100% of I. We take a portion of I by working in a zig-zag order, as in the `jsteg` algorithm.

DCT historgrams

In [10] Fridrich devised a set of statistical features designed to detect the steganographic strategies behind the `F5` algorithm, among others. We selected 5 of those features for use in our neural nets.

Called *calibration,* the process is similar to that used with the image quality metrics, but is specifically designed for jpeg images. Figure 5 illustrates the process. A jpeg image (I) is first decompressed back to a bitmap. The bitmapped image is cropped by four pixels along each edge. Then the bitmap is recompressed to a jpeg (I_c), using the same compression ratio as the original image. The statistics then measure the difference in the distribution of DCT coefficients between I and I_c.

In each of the following equations, the symbol $\|x\|_{L_1}$ denotes $\sum_i |x_i|$, called the L_1-norm of the vector x. Also $\mathcal{N}(x) = x/\|x\|_{L_1}$ is the normalization of the vector x.

The *global histogram feature,* equation (7), compares the global distribution of the quantized DCT coefficient values of I to the distribution of I_c.

$$F_1 = \|\mathcal{N}(H(I)) - \mathcal{N}(H(I_c))\|_{L_1}, \tag{7}$$

where H is the frequency count of an image's quantized DCT coefficients' values.

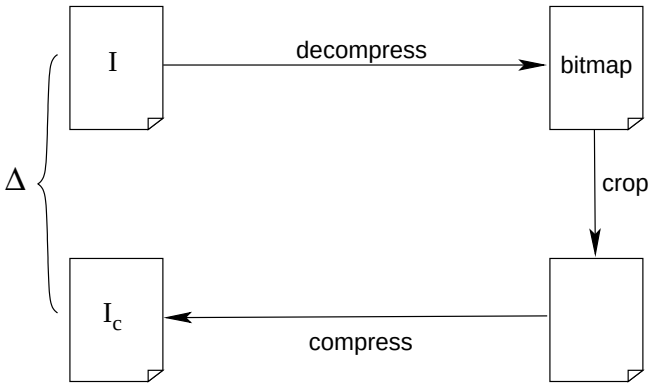

Figure 5: The DCT calibration technique. I denotes a jpeg image, I_c its calibrated counterpart.

The *individual histogram feature for (1,3),* equation (8), compares the distribution of quantized DCT coefficients in the (1,3) position in each 8×8 block.

$$F_2 = \left\| \mathcal{N}(h^{13}(I)) - \mathcal{N}(h^{13}(I_c)) \right\|_{L_1}, \tag{8}$$

where h^{13} is the histogram of an image's quantized DCT coefficients in the (1,3) position of each 8×8 block.

The *dual histogram feature for 0,* equation (9), compares the number of quantized DCT coefficients equal to 0 in each position of the 8×8 blocks.

$$F_3 = \left\| \mathcal{N}(G^0(I)) - \mathcal{N}(G^0(I_c)) \right\|_{L_1}, \tag{9}$$

where G^0 is the 8×8 matrix of elements $g^0_{i,j} = \sum_k \delta\big(0,\, d_k(i,j)\big)$, $d_k(i,j)$ is the value in the (i,j) position of the k^{th} block, and δ is the Dirac delta function.

The L_1 *blockiness feature,* equation (10), employs the blockiness statistical test which compares pixel values along the edges of 8×8 blocks.

$$F_4 = \left\| \mathcal{N}(B(I)) - \mathcal{N}(B(I_c)) \right\|_{L_1}, \tag{10}$$

where

$$B = \frac{\sum_{i=1}^{M'} \sum_{j=1}^{N} |x_{8i,j} - x_{8i+1,j}| + \sum_{j=1}^{N'} \sum_{i=1}^{M} |x_{i,8j} - x_{i,8j+1}|}{N \cdot M' + M \cdot N'}$$

and $x_{i,j}$ is the value of the pixel in the (i,j) location, the dimensions of the image are $M \times N$ and $M' = \lfloor (M-1)/8 \rfloor$, $N' = \lfloor (N-1)/8 \rfloor$.

The *Co-occurrence feature for (1,1),* equation (11), compares the values of quantized DCT coefficients in neighboring blocks.

$$\begin{aligned} F_5 =& \big(C_{1,1}(I) + C_{1,-1}(I) + C_{-1,1}(I) + C_{-1,-1}(I)\big) - \\ & \big(C_{1,1}(I_c) + C_{1,-1}(I_c) + C_{-1,1}(I_c) + C_{-1,-1}(I_c)\big) \end{aligned} \tag{11}$$

where

$$\begin{aligned} C_{s,t} =& \frac{32}{MN} \sum_{k=0}^{(M-2)/8} \sum_{l=0}^{(N-1)/8} \sum_{i,j=1}^{8} \delta(s,\, d(8k+i, 8l+j))\delta(t,\, d(8(k+1)+i, 8l+j)) \\ & + \frac{32}{MN} \sum_{k=0}^{(M-1)/8} \sum_{l=0}^{(N-2)/8} \sum_{i,j=1}^{8} \delta(s,\, d(8k+i, 8l+j))\delta(t,\, d(8k+i, 8(l+1)+j)), \end{aligned}$$

and $d(i, j)$ is the quantized DCT coefficient value in the (i, j) location.

3.3 Design of Artificial Neural Networks

An *artificial neural network* is a nonlinear optimization technique that is used to predict the behavior of complex systems. It can also be used, as in this project, to perform pattern recognition and data classification. Data is presented to the ANN as a set of points in a high-dimensional space. Each point is labeled as belonging to one of two classes: A or B. The ANN determines a nonlinear function that separates the A-points from the B-points. This function can then be used to predict the class of new data points whose character was heretofore unknown.

One particular merit of an artificial neural network is that it is adaptive—as additional data is provided to the system it refines its prediction function. In this way the pattern recognizer can respond to evolution in the data. For example, if small modifications are made to an existing steganographic algorithm, the software will be able to adapt.

The Adaptive Computing Laboratory (ACL) Toolkit (henceforth, Toolkit) is a comprehensive system developed at Iowa State University for advanced artificial neural network development and application. The Toolkit is an open-source software package available from [2]. We selected this software for our pattern recognition system because of its extensive and sophisticated proven capabilities at solving difficult data mining problems [3, 19].

The set of weight values and number of nodes for an ANN is called the ANN architecture. The Toolkit allows the ANN architecture to be simultaneously and automatically optimized using a dynamic node approach, an advanced technique not available in commercial ANN software. The number of optimal nodes is determined by the ANN itself using information gleaned from the data. The weight

values are optimized using a scaled conjugate gradient descent learning algorithm (SCGD). This learning algorithm is faster than the back-propagation learning algorithm by a factor of 1000 or more, and thus the ANN trained with SCDG can attempt bigger problems such as processing with image data with reasonable computer resources. Thus, with a better modeling capability than previously used ANNs, we have achieved initial results that outperform previous published results using ANNs [4, 17].

During the course of this project we discovered that we obtained much better results if we developed three separate networks, one for each of the three target stego algorithms.

3.4 Implementation and Testing

The front-end for the package was written in Visual Basic. The mathematical functions are written (and compiled) in Matlab 7.0.1. the Matlab Component Runtime Library serves as the interface between the front- and back-ends. The front-end provides an easy-to-use graphical user interface that is similar to other Microsoft Windows applications. Matlab is an excellent development platform for this sort of application. It provides ready access to a large library of advanced mathematical functions (such as the discrete cosine transform) and makes array-handling very intuitive. The downside is that execution time and memory use are much worse than they would be for a natively-compiled application. In the long run it will be necessary to reimplement the package in a low-level language, but that is outside the scope of this project.

The software was tested by applying the neural nets to approximately 250 images, half stegograms and half innocent. This was done for each of the three steganographic algorithms at both 50% and 100% of capacity. The results are described in Section 4.

The neural net compiles the features for an image and returns a score with a target of s_0 for innocent and s_1 for stegogram. A threshold t is chosen (generally, $t = \frac{1}{2}(s_0 + s_1)$) and any image whose score is below the value t is judged to be innocent. Those with scores above t are predicted to be stegoimages. Figure 6 shows the output of the ANN for one run of 244 images. Images 1–122 all contain data hidden with `jsteg` at 50% of capacity. (In this instance, $s_0 = 0$, $s_1 = 1$ and $t = .5$). Observe that every one of them has a score above t, so the ANN predicts correctly in every case. This is reflected in the 100% true positive entry in Table 1. Images 123–244 are all innocent. Note that all but two of them have scores lying below the 0.5 threshold. Thus the ANN had an approximately 2% false positive rate for this algorithm.

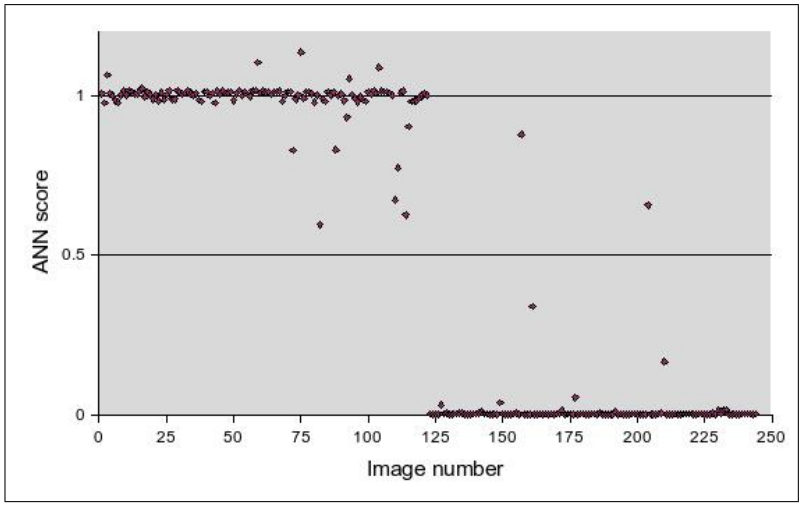

Figure 6: Graph of output values from ANN for a set of 244 images

	100% of Capacity		50% of Capacity	
Stego algorithm	True Pos	False Pos	True Pos	False Pos
jsteg	97%	0%	100%	2%
F5	92%	5%	85%	20%
wavelet	95%	16%	97%	16%

Table 1: Results of ANN classification. **True Pos** denotes the percentage of stego images correctly identified as such. **False Pos** is the percentage of innocent images incorrectly identified as having hidden data.

4 Results and Discussion

ANNTS is quite successful at identifying all stegoimages, although the results varied depending on the particular steganographic algorithm and amount of hidden data. As Table 1 shows, our distinguisher is virtually flawless at recognizing data hidden using `jsteg`. This is largely due to a known weaknesses in the jsteg algorithm that the balanced parity feature (section 3.2) is designed to exploit. ANNTS misses at most 3% of all images embedded to capacity with `jsteg`, and incorrectly flags an innocent image as having even a small amount of hidden data at most 2% of the time.

Our techniques were somewhat less successful against the `F5` algorithm, which

was specifically designed as an improvement to jsteg. However, at least for stegoimages with data embedded at close to full capacity our methods are quite successful. ANNTS caught 92% of all images embedded to full capacity with `F5` and misidentified only 5%. Unfortunately our results at the 50% of capacity level are less satisfactory. In particular, the system falsely labels 20% of all innocent images as containing hidden data. There are several possible explanations for this. The detection rate may be dependent on the nature of the image being examined. For example, ANNTS may be more effective on photographs (which have many variations in color or shade) than on computer-generated images or images with large uniform areas such as sky. Further research is necessary on this point, but the results are encouraging.

Finally, we were able to identify correctly 96% of the stegoimages created with our wavelet embedding algorithm, with a false positive rate of 16%. To our knowledge we are the first to perform tests on embeddings in the wavelet domain. Certainly a 16% false positive rate is unacceptably high and requires further research.

4.1 Comparison to other research

It is desirable to compare our results to those obtained by others in the field. Unfortunately, this is not easy to do since thus far each research team has used a different testing methodology.

Westfeld and Pfitzmann [22] perform some steganalysis on `jsteg` but not an extensive enough set of tests to form a basis of comparison. Lyu and Farid [12] use a one-class support vector machine in a manner similar to our artificial neural nets. While their true positive results for `jsteg` and `F5` are not as strong as ours, it is not clear to what degree the stegoimages contained embedded data up to the capacity of the cover. Also, they seemed to insist on a very low false positive rate. For example, for `F5` they obtain a true positive rate of 51% with a false positive rate of 1%.

In [10], Fridrich performed tests similar to ours using a Fisher linear discriminant instead of an artificial neural net. She describes her results in terms of the area under the ROC curve. This has the advantage that it combines both the true and false positives in a single measure. The ROC curves for our tests on `F5` are shown in Figure 7. More precisely, Fridrich uses the "detection reliability" measure $\rho = 2A - 1$, where A denotes the area under the ROC curve. We obtain values of 0.74 and 0.95 for ρ (at 50% and 100% of capacity) compared to 0.96 and 0.99 for Fridrich.

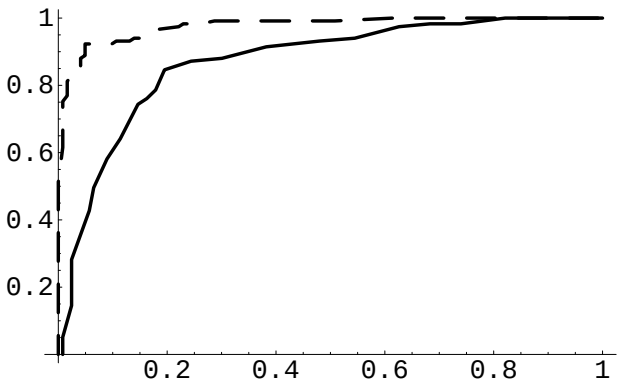

Figure 7: ROC curve for detection of F5-steganography at 50% of capacity (solid line) and at 100% of capacity (broken line)

4.2 The user interface

While our package is not a polished piece of commercial software, we believe it will be easy to install and use on any reasonably recent PC running Microsoft Windows. After starting the program, the user is presented with a window as in Figure 8. The selection dialog in the top center of the window can be used to choose a directory containing the images to be scanned. In the center-right appears a list of images. These can be previewed in the bottom left portion of the window.

Once an image is selected, the user clicks the "Analyze Selected Image" button. There is a delay while the computations are performed. When complete, the software will indicate whether it predicts the existence of hidden messages using the check boxes in the center of the screen. In the example illustrated above, the software concludes that the image falls in "class 4". That is, data hidden at the 50% of capacity level using the F5 algorithm.

5 Dissemination

On July 28, 2006 we demonstrated the software package at a meeting of the *Iowa Internet Crimes Against Children Task Force.* About 10 agents were present, including our host, Special Agent Supervisor Michael Morris. The agents made several suggestions to enhance the utility of the software. Some of these are quite straightforward to implement, and we have done so for the final release of the software. Others are much more involved and will have to wait for a future project to be implemented.

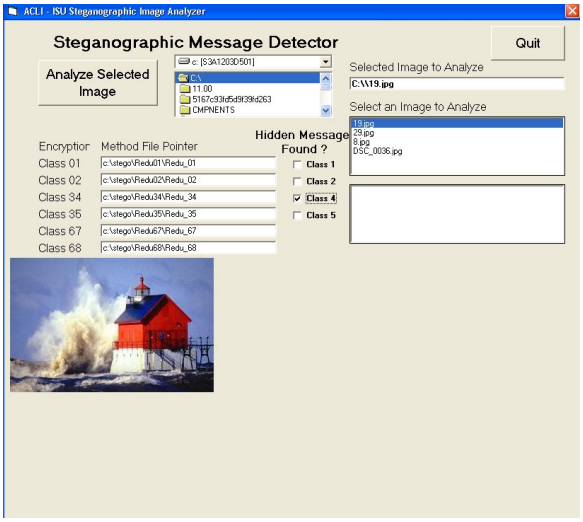

Figure 8: Screen shot from the ANNTS software

Through Mr. Morris, we will provide the ICAC (as well as the MFRC) with copies of the software and a brief users manual. In addition we are preparing a paper for publication, hopefully in the IEEE Journal of Forensic Science. Preliminary results were published and presented at the Annual SPIE meeting in San Diego, 2005 [7].

We are pleased to report that the software has had its first field test, which was successful. Detective David Schwindt of the Iowa City Police Department contacted us with an image he wished to have analyzed. ANNTS returned a negative response. Subsequently Detective Schwindt uncovered the original source of the photograph and determined that it was identical (both images had the same SHA1 hash) to the version under investigation. Thus our conclusion that the image was innocent appears to be correct.

APPENDIX

B The Wavelet Embedding Algorithm

The "wavelet embedding" applied here is not a true steganographic algorithm. For one thing, the extracted payload contains far too many errors to be useful in practice. For another, the stegoimage is a bitmap, which is not very realistic. Conversion to jpeg or jpeg2000 would undoubtedly destroy the payload.

Briefly, a 3-scale integer-to-integer wavelet transform is applied to a grayscale image in bitmap form. The transform used was the $(4, 2)$ integer-to-integer interpolating transform of Calderbank, Daubechies, Sweldens and Yeo [5]. The one-dimensional transform of a sequence $s_{0,*}$ is given by the equations

$$d_{1,l} = s_{0,2l+1} - \left\lfloor \frac{9}{16}(s_{0,2l} + s_{0,2l+2}) - \frac{1}{16}(s_{0,2l-2} + s_{0,2l+4}) + \frac{1}{2} \right\rfloor$$

$$s_{1,l} = s_{0,2l} + \left\lfloor \frac{1}{4}(d_{1,l-1} + d_{1,l}) + \frac{1}{2} \right\rfloor$$

where $d_{1,*}$ denotes the high-frequency component and $s_{1,*}$ the low-frequency component of the transform. The two-dimensional transform is separable, thus it is obtained by applying the above equations first to the rows and then the columns of an array. It is important to note that unlike the discrete cosine transform, every component of this transform is again an integer, although it may not lie in the interval $0 \ldots 255$.

When the transform is applied to an image of size $n \times m$, the result is 4 "images" of size $\frac{n}{2} \times \frac{m}{2}$ consisting of the high/high, high/low, low/high and low/low components of the transform. In a 3-scale transform, the transform is applied a second time to the low/low component and then a third time to the low/low component of the second scale..

The embedding strategy is very simple. The least significant bit of a wavelet coefficient is replaced by one bit of the payload. This means that a coefficient is

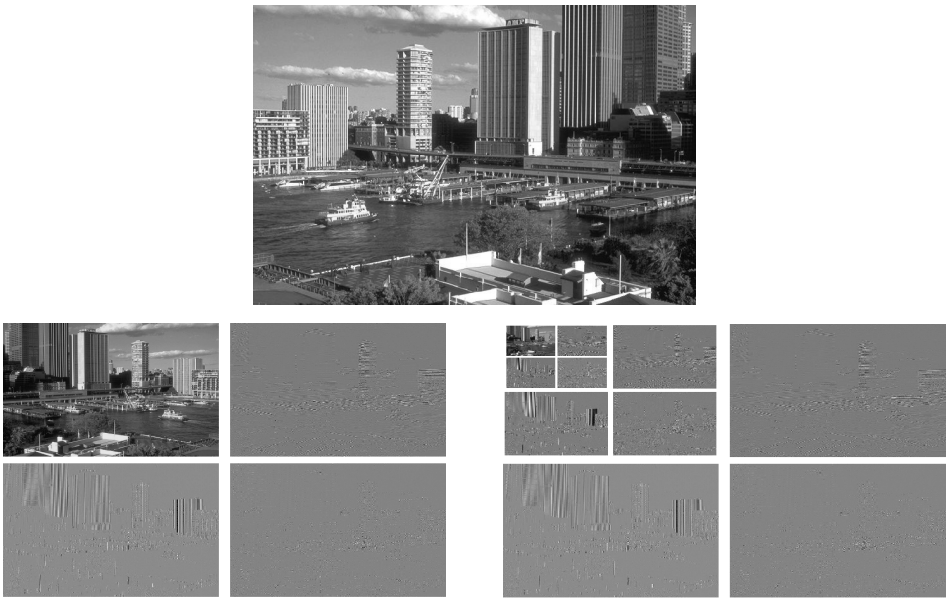

Figure 9: An image, its 1-scale and 3-scale wavelet transforms. In the middle image, the low/low component is at the top left, the high/high component is the bottom right.

changed by at most one, and on average, a coefficient changes by about .5. This replacement is applied to every coefficient in the transformed image *except* those in the low/low component of the third-level wavelet. (In Figure 9 this exception is the little image in the very top-left corner of the 3-scale transform.) Thus, if the original image has n pixels, there are $n/4^3$ coefficients that are left alone.

After embedding, the inverse transform is applied. The output will be a rectangular array of integers, however, there will typically be values outside the range $[0, 255]$. These are clipped, so that negative values are replaced by 0 and values above 255 are replaced by 255. It is the clipping that introduces errors upon extraction.

Despite its shortcomings, we believe this technique is useful for steganalytic purposes. We desire statistical techniques that will determine whether an image has been modified by a perturbation of its wavelet transform. Since this embedding algorithm does that, it can serve as a reasonable testbed for candidate statistics.

C Final Feature Set

During the training phase of the project we pared the original set of 81 features (discussed in Section 3.2) down to 22 that showed the most promise for steganalysis. We list them here.

mean(3,H)
variance(2,H)
variance(3,H)
kurtosis(2,D)
mean error(2,V)
mean error(3,H)
mean error(3,V)
mean error(3,D)
variance error(3,H)
variance error(3,V)
skew error(2,H)
skew error (3,H)
skew error (3,V)
kurtosis error(3,V)
balanced parity (equation (6))
global histogram (equation (7))
individual historgram for $(1, 3)$(equation (8))
dual histogram for 0 (equation (9))
L_1 blockiness (equation (10))

co-occurence for $(1, 1)$ (equation (11))
median weighted block spectral distortion (equation (4))
normalized HVS metric (equation (5))

References

[1] I. Avcibaş, N. Memon, and B. Sankur. Steganalysis using image quality metrics. *IEEE Trans. on Image Processing*, 12(2):221–229, 2003.

[2] E. Bartlett. The adaptive computing laboratory toolkit. available at http://www.public.iastate.edu/~ebart, 2003. Iowa State University.

[3] E. B. Bartlett and A. Whitney. On the use of various input subsets for stacked generalization. In Dagli et al., editors, *Artificial Neural Networks in Engineering*, volume 9 of *Intelligent Engineering Systems Through Artificial Neural Networks*, pages 111–116, St. Louis, Missouri, Nov. 1999. American Society of Mechanical Engineers.

[4] G. Berg et al. Searching for hidden messages: Automatic detection of steganography. In *Proc. 15th Innovative Applications of Artificial Intelligence Conf.*, pages 51–56, Acapulco, Mexico, 2003.

[5] R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo. Wavelet transforms that map integers to integers. *Applied and Computational Harmonic Analysis (ACHA)*, 5(3):332–369, 1998.

[6] J. Daugman. Entropy reduction and decorrelation in visual coding by oriented neural receptive fields. *IEEE Trans. Biomedical Engineering*, 36(1):107–114, 1989.

[7] J. Davidson, C. Bergman, and E. Bartlett. An artificial neural network for wavelet steganalysis. In *Optics and Photonics*, volume 5916 of *Mathematical Methods in Pattern and Image Analysis*, pages 1–10. SPIE, 2005.

[8] E. Delp. Watermarking evaluation testbed. http://www.datahiding.org. courtesy of Dr. Edward Delp, Purdue University.

[9] H. Farid. Detecting hidden messages using higher-order statistical models. In *Proc. International Conference on Image Processing*, volume 2, pages 905–908, 2002.

[10] J. Fridrich. Feature-based steganalysis for jpeg images and its implications for future design of steganographic schemes. preprint, 2004.

[11] G. Krieger, C. Zetzsche, and E. Barth. Higher-order statistics of natural images and their exploitation by operators selective to intrinsic dimensionality. In *Higher-Order Statistics*, Proceedings of the IEEE Signal Processing Workshop, pages 147–151, July 1997.

[12] S. Lyu and H. Farid. Steganalysis using color wavelet statistics and one-class support vector machines. In *SPIE Symposium on Electronic Imaging*, 2004. San Jose, CA.

[13] A. McAdams and T. McKay. Steganalysis: an exploration of methods. REU project, Iowa State University, 2005.

[14] N. Nill. A visual model weighted cosine transform for image compression and quality assessment. *IEEE Trans. Communications*, com-33(6):551–557, June 1985.

[15] Veiled messages of terrorists may lurk in cyberspace. New York Times, Oct. 30 2001.

[16] N. Provos and P. Honeyman. Hide and seek: an introduction to steganography. *IEEE Security & Privacy Magazine*, 1(3):32–44, 2003.

[17] L. Shaohui, Y. Hongxun, and G. Wen. Neural network based steganalysis in still images. In *Proc. Intl. Conf. on Multimedia and Expo, ICME 2003*, volume 2, pages 509–512, 2003.

[18] G. Simmons. The prisoners' problem and the subliminal channel. In *Advances in cryptology (Santa Barbara, Calif., 1983)*, pages 51–67. Plenum, New York, 1984.

[19] D. V. Sridhar, E. B. Bartlett, and R. C. Seagrave. Information theoretic subset selection for neural network models of chemical processes. *Computers and Chemical Engineering*, 22(4/5):613–626, 1998.

[20] D. Upham. jsteg. available from http://stegoarchive.com.

[21] A. Westfeld. F5—a steganographic algorithm. In I. Moskowitz, editor, *Information Hiding: 4th International Workshop*, volume 2137 of *Lecture Notes in Computer Science*, pages 289–302, Berlin, 2001. Springer-Verlag.

[22] A. Westfeld and A. Pfitzmann. Attacks on steganographic systems. In *3rd Int'l Workshop on Information Hiding*, volume 1768 of *Lecture Notes in Computer Science*, pages 61–75, Berlin, 2000. Springer-Verlag.